INTL-0606-US
(P11747)

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: CACHE ARCHITECTURE TO REDUCE LEAKAGE POWER CONSUMPTION

INVENTOR: DENNIS M. O'CONNOR

Express Mail No. EL911616190US

Date: August 13, 2001

CACHE ARCHITECTURE TO REDUCE LEAKAGE POWER CONSUMPTION

Background

This invention relates to the caches, including the L1 or level 1 and L2 or level 2 caches normally associated with microprocessors.

Conventional microprocessor architecture schemes use an L1 and an L2 cache to temporarily store instructions, state information, functions, and other information.

The level 1 instruction caches service requests for instructions generated by instruction prefetchers. The level 1 data cache caches service memory data read and write requests generated by the processor's execution units when they are executing instructions that require a memory data access.

The level 2 cache resides on the dedicated bus and services misses on the level 1 cache. In the event of a level 2 cache miss, the level 2 cache issues a transaction request to an external bus unit to obtain the requested instruction or data line from external memory. The information is placed in the level 2 cache and is also forwarded to the appropriate level 1 cache for storage.

When the prefetcher requests a line of code from a code cache, the request results in a hit or a miss. In the event of a miss, the code cache issues a request to the

level 2 cache. A look-up is performed in the level 2 cache indicating a hit or a miss. In the case of a hit, the requested line is supplied to the code cache. If the request results in a level 2 cache miss, the level 2 cache issues a request and the line is read from external memory.

As semiconductor devices become smaller and smaller, leakage power consumption considerations become more and more important, especially for mobile applications. As a result, leakage power consumption may become a significant contributor to total power dissipation.

Thus, there is a need for ways to design multilevel caches to reduce cache leakage power consumption.

Brief Description of the Drawings

Figure 1 is a schematic depiction of a multilevel cache in accordance with one embodiment of the present invention;

Figure 2 is a schematic depiction of a processor-based system in accordance with one embodiment of the present invention;

Figure 3 is a flow chart for software for handling a read miss in accordance with one embodiment of the present invention; and

Figure 4 is a flow chart for software for maintaining cache coherency in accordance with one embodiment of the present invention.

Detailed Description

In accordance with some embodiments of the present invention, power dissipation may be reduced by placing frequently used, time critical functions and state information in a first level cache containing relatively fast components that necessarily have relatively higher leakage currents. Other functionality may be migrated to a second level cache made up of slower components that have lower leakage current. The functionality that remains in the faster, higher leakage components may be referred to herein as the core.

In the cache hierarchy then, functions that remain in the core may include things such as tags, valid bits and the data itself. In some embodiments, the core may include debug and analysis and trace flags, as well as access control attribute bits. In one embodiment, virtual addresses may be utilized to index the core to avoid the need for an address translation mechanism, such as a translation look aside buffer (TLB). This use of virtual addressing may reduce the amount of state in the core and the number of nodes that are toggled during instruction execution.

In addition, the L1 caches may be write-through to reduce complexity and to enable certain functions to be performed in the L2 cache. In some embodiments, line

replacement policy may be implemented by the L2 instead of the L1 cache.

Management of the L1 cache may be implemented by the L2 cache or caches implemented in slower devices with lower leakage currents. In addition to the usual L2 mechanisms, the L2 cache may contain mechanisms for managing L1 cache line replacement, performing virtual-to-physical translation, ensuring L1 cache coherency and determining the access attributes of memory regions.

Sub A1

Referring to Figure 1, the L1 cache 12 may be connected by a high bandwidth link to the L2 cache 14. In accordance with one embodiment of the present invention, the L2 cache may be a unified L2 cache. In another embodiment of the present invention, a single core or two more cores may be utilized in systems with separate L2 caches for instructions and data. The L1 cache 12 may include an instruction cache 16, a pipeline 18 and a data cache 20. Two separate L1 caches 14a and 14b may be provided in one embodiment. As a result, cache management logic, snooping support, debugging and monitoring mechanisms and virtual-to-physical translation may be removed from the L1 caches while still supporting, by a mechanisms in the L2 cache, L1 cache coherency, trace, breakpoints, performance monitoring and virtual memory in the L2 cache 14 as indicated in block 22.

As a result, the L2 cache 14 can be made simpler, may be faster and may be more energy efficient because only the higher leakage current components that are needed are utilized and all other functions are diverted to a lower leakage L2 cache 14.

Referring next to Figure 2, a processor-based system 60 may include an integrated circuit 62 that includes the L1 cache 12 as well as the L2 cache 14 in one embodiment. Register 64 may be coupled to the L1 cache 12. The integrated circuit 62 communicates with a random access memory (RAM) 66. Software 50 and 26 may be stored in the RAM 66. The input/output (I/O) bus 68 communicates with the RAM 66 and the integrated circuit 62 through the system bus 70.

Referring to Figure 3, the software 26 for handling a L1 read miss via the L2 cache 14 is illustrated. On a L1 cache read miss, an L1 cache 12 passes the details of the access to the L2 cache as indicated in block 28. These details may include, for example, the type, size, virtual address and destination register.

The L2 cache 14 may then use its memory translation mechanism, such as a translation look aside buffer to determine the physical address and attributes of the access as indicated in block 30. The L2 cache may also check to see if the requested data is in the L2 cache 14 as determined in diamond 32. If the attributes indicate

access, the requested data may be fetched from memory (either from the L2 cache or from further out in the memory hierarchy) as indicated in block 36. If the data is in the L2 cache 14, it may be fetched from the L2 cache 14 as indicated in block 34.

A check at diamond 38 determines whether the access was cacheable. The L2 cache 14 ensures that the data is cached in the L2 cache 14 and then fetches data the width and alignment of an entire L1 cache 12 line and returns the data to the L1 cache as indicated in blocks 42 and 44. Along with the data, information may be sent to the L2 cache 14 as indicated above. The access attributes are read from a translation look aside buffer, any relevant breakpoint, performance monitoring and trace tags, the way within the set to store the line into and a signal indicating that the indicated way in the appropriate set should be replaced with the data and tags given as indicated in block 44. The information indicating which way the data was stored into the L1 cache is also recorded in the corresponding line of the L2 cache for future use as indicated in block 46. If there are multiple L1 caches served by the same L2 cache, storage for the L1 cache location information may be available for each L1 cache.

There are a number of ways that the L2 cache may determine which way of the L1 to replace. A pseudo-random scheme may be used or there may be a mapping or partial

mapping between which L2 cache way contains the data and which L1 cache way contains the data, or any number of L2 or possibly even L1 access or replacement history schemes may be used in other embodiments.

If the access was not cacheable as determined in diamond 38, and the access is legal, the L2 cache may retrieve exactly the requested data. The requested data may be returned to the L1 cache as indicated in block 40 along with the original information sent to the L2 cache and a signal indicating that the data should not be stored in the L1 cache.

All data loaded into the L1 instruction caches are executable. The only attributes that the L2 instruction cache stores are those related to breakpoint, trace, or performance monitoring events, in one embodiment.

There is no need to record in the L1 data cache whether the memory region that the line is mapped into is cacheable or not. Depending on whether there are write buffers in the core, where the region is bufferable may or may not be one of the attributes stored in the L1 cache. Whether the memory can be written to or not is an attribute stored in the cache. Flags that indicate that a trace, performance counter, or breakpoint event should occur may be part of the L1 cache attributes. The granularity of these flags (one per line, one per word, or some other

scheme) and other specifics are architecture and implementation dependent.

Turning to Figure 3, the L2 cache can snoop the bus leading further out in the memory hierarchy. In addition, since all L1 caches served by the L2 cache are write-through, the L2 cache sees all modifications made by the core it serves. Since the L2 cache is inclusive (all valid lines in the L1 caches have corresponding valid lines in the L2 caches), any change to memory cached in an L1 cache is also a change to memory cached in the L2 cache.

When the L2 cache notes a change to its contents, the L2 cache checks the affected line to see which, if any, of the L1 caches also have the address cached. The L2 cache then uses the information it has stored about which way each L1 cache is using to store the information, and sends each affected L1 cache an address, a way designator, and a signal indicating that that way of that set should be invalidated. Alternatively, the L2 cache sends an address, a way, the new data and its size and a signal indicating that the data supplied should be written into the appropriate set in the indicated way.

The L1 cache may be virtually indexed and the L2 cache may be physically indexed. The index is the same for both in some embodiments. The mapping within each L2 cache line of which L1 cache way within each set holds the data in that L2 line serves as a physical-to-virtual address

translation. Thus, if two cores were served by the same L2 cache, and each was using different address mapping so that each was accessing the same physical address through two different virtual addresses, both would still be properly updated to maintain cache coherence.

Thus, referring to Figure 4, the software 50 determines whether there is a change of contents at diamond 52. If so, the identity of the L1 caches that have the address are determined as indicated in block 54. The affected L1 cache address information, way designator and a signal are sent (block 56).

While the present invention has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.

What is claimed is: